FAME-PERMIS Project Output

WORKPACKAGE 7 – Deliverable D10

The FAME System User Guide

Aleksandra Nenadić
Ning Zhang

School of Computer Science
University of Manchester

March 2007

The FAME System User Guide: Configuration and Installation

Table of Contents

FAME User Guide: Configuration and Installation

This user guide explains how to install and configure the FAME module in an Apache2-Perl Web Server run by a Shibboleth IdP (Identity Provider). The FAME module integrates multiple authentication services provided by the IdP. It authenticates users, derives a LoA (Levels of Assurance) based upon the authentication service and token used in an authentication instance, and passes the derived LoA values to SPs (Service Providers) via the Shibboleth protocol so as to achieve LoA-linked fine-grained access control in the Shibboleth infrastructure.

The FAME module, *Fame.pm,* is configured via directives placed in the Apache configuration file, *httpd.conf*). So this document should be read in conjunction with the standard Apache Web Server documentation (available at http://httpd.apache.org/docs/2.0/). The FAME module also makes use of an external relational database to store secret cryptographic keys, and information about users and Authentication Servers run at the IdP (more details are given in Section 2 below).

It is assumed that you have already had a working installation of a Shibboleth IdP. That is, you have already installed an Apache Web Server, Tomcat servlet container, mod_jk Tomcat-Apache plug-in that handles the communication between the Tomcat container and the Apache Web Server, Shibboleth IdP, and LDAP Directory that stores Shibboleth attributes.

0 FAME Prerequisites and Installation Kit

The configuration/installation instructions described in this document are given with reference to the Linux/Unix environment. Before you start to install and configure FAME, the following gives a detailed list of the required prerequisite components:

(1) Apache Web Server version 2.0, available from http://apache.org/.
(2) Perl interpreter (version 5.8.6 was used for the development of FAME), available from http://www.perl.com/.
(3) mod_perl – an Apache module for providing a persistent Perl interpreter embedded in the Web Server for creating Apache-Perl modules, available from http://apache.perl.org/.
(4) The following Apache-Perl modules are required for the correct functioning of FAME: Apache2::Request, Apache2::RequestRec, Apache2::RequestIO, Apache2::RequestUtil, Apache2::ServerRec; Apache2::ServerUtil, Apache2::Connection, Apache2::Log, Apache2::Const, Apache2::Cookie, APR::Table, APR::Const, ModPerl::Registry, ModPerl::Util, Crypt::CBC, Crypt::Rijndael, Crypt::Random, Digest::MD5, MIME::Base64, IO::File, DBI, Net::LDAP. All these modules are available from http://www.cpan.org/.
(5) Mysql database version 4.1 or above, or a similar relational database system.

We have used Shibboleth IdP version 1.3 when developing the FAME system, which is available from http://shibboleth.internet2.edu/.

The FAME installation kit comprises the following items:

(1) The source code, *Fame.pm,* of the Apache-Perl FAME module.

(2) The *fame* folder containing two sub-folders, *images* and *css,* which contains, respectively, images and a style-sheet, used by the FAME module for rendering HTML pages.
(3) The database set-up script, *fame.sql,* which can be used to create the FAME database.
(4) An exemplar Apache-Perl script, *AS.pm,* for setting up an Authentication Server in conjunction with the Apache Web Server.
(5) An LDAP schema file, *fame.schema,* for integrating FAME introduced attributes with the Shibboleth's LDAP store.
(6) A report describing the design of the FAME system (it is advised that you read this design document before starting to configure FAME).
(7) FAME User Guide, i.e. this document.

In order to configure a Shibboleth IdP to use FAME, the following tasks should be performed:

(1) Configure Apache2 Server to use the FAME module, *Fame.pm*.
(2) Create the FAME database
(3) Set up the Authentication Server(s) and configure FAME to use them.
(4) Integrate FAME and Shibboleth.

The following four sections address each of the above tasks, respectively.

1 Configuring FAME and Apache

The FAME module has been developed for and tested with Apache2 under the assumption that an Apache2 Server with a support for Perl (via mod_perl) has been previously installed. The FAME system is implemented as an Apache-Perl module, called *Fame.pm* (the source code can be found in the Installation Kit). The module is created under the *MyApache2*:: namespace, thus the full name of the FAME module is *MyApache2::Fame.pm*.

Locating the module
First you should create a directory where the FAME module will reside in. For example, you may create a directory called *perl* within your Web Server root directory (which, on our system, is */etc/apache2*), where you will keep all your Perl modules, i.e. create a directory */etc/apache2/perl*. As the module is created under the *MyApache2*:: namespace, create the *MyApache2* subdirectory within the *perl* directory. The *MyApache2* subdirectory is where the *Fame.pm* module should be located.

Next, you need to tell Apache where to look for *Fame.pm*. Apache's *mod_perl* can be configured to invoke a start-up file (typically called *startup.pl*) containing a set of Perl commands, each time the server is launched or restarted. This is where we place the *'use lib*' statement that will instruct Apache where to find *Fame.pm*. You may also include configuration directives for the FAME module in *startup.pl* after the '*use lib*' command. Alternatively, these configuration directives can be configured via *httpd.conf*. If you choose to configure the module in *startup.pl*, the following snap-short gives an example of how this can be done (the commands related to the FAME module are placed at the bottom of the snap-short and highlighted in green). Place *startup.pl* in the directory containing all other configuration files for the Apache

modules (i.e. files ending with *.conf*). The meanings of the FAME configuration directives used in this example will be explained shortly.

File: /etc/apache2/modules.d/startup.pl

```
use lib qw(/home/httpd/perl);

use ModPerl::Util (); #for CORE::GLOBAL::exit

use Apache2::RequestRec ();
use Apache2::RequestIO ();
use Apache2::RequestUtil ();

use Apache2::ServerRec ();
use Apache2::ServerUtil ();
use Apache2::Connection ();
use Apache2::Log ();

use APR::Table ();

use ModPerl::Registry ();

use Apache2::Const -compile => ':common';
use APR::Const -compile => ':common';

# FAME-related commands
# Location of the FAME module
use lib '/etc/apache2/perl';

# Load FAME module before configuring it
use MyApache2::Fame ();

# Configure FAME module
Apache2::ServerUtil->server->push_handlers(PerlChildInitHandler =>
\&MyApache2::Fame::configure((
        'FameLoginServer' => '/fame_login_server',
        'FameLogoutHandler' => '/fame_logout',
        'FameAuthTimeout' => 480,
        'FameSSOTimeout' => 1,
        'FameDB' =>
'dbi:mysql:database=fls;host=localhost;port=3306',
        'FameDBUser' => 'flsuser',
        'FameDBPassword' => 'flspassword',
        'FameSecretsTable' => 'secrets:secret_key:date_time',
        'FameASTable' =>
'authservers:url:auth_type:saml_auth_id:loa:secret_key',
        'FameUsersTable' =>
'fameusers:ldap_id:ldap_attribute:alternative_id',
        'FameShibLDAPServer' => 'localhost',
        'FameShibLDAPPort' => '389',
        'FameShibLDAPDN' =>
'cn=Manager,dc=rpc56,dc=cs,dc=man,dc=ac,dc=uk',
        'FameShibLDAPPassword' => 'lm^s7a*56',
        'FameShibLDAPBaseDN' =>
'ou=shib-users,dc=rpc56,dc=cs,dc=man,dc=ac,dc=uk'
        )));

1;
```

Finally, copy the *fame* directory from the installation kit, containing folders *images* and *css*, to your Apache Web Server's document root (on our system it is */var/www/localhost/htdocs*). It is important to copy these directories in your Web Server's document root as they have to be "visible" from the Web; they will be used by the FAME module for generating its HTML pages.

Loading the module

In order to configure the FAME module, *Fame.pm,* for it to work with the Apache Web Server, the module must first be uploaded before using any of the directives explained below. If you have not uploaded the module in *startup.pl* as previously explained, then do so in Apache's configuration file, *httpd.conf. That is,*

File: /etc/apache2/httpd.conf

```
...
# Load FAME module before configuring it
use MyApache2::Fame ();
...
```

Configuration directives

In order to configure the FAME module, the following three blocks should be created and configured through the use of various FAME directives in the Apache's configuration file, *httpd.conf*. The first block specifies the F-SSO (FAME's Single Sign On) checker as the access control handler for the location (i.e. url) of the Shibboleth's HS (Handle Service). By doing so, F-SSO is set up to protect the url of the Shibboleth's HS and AuthType (the authentication type) for this location is set to 'Fame'.

The following gives an exemplar setting of this block. The block specifies that the *Apache2::Fame.pm* module's *sso_checker( )* routine is responsible for user authentication before the users are allowed to access the Shibboleth HS's url (tied to location /shibboleth-idp/SSO). The routine will intercept all the requests for accessing the Shibboleth HS, and redirect any request that does not already contain an SSO cookie to the F-LS (FAME Login Server).

File: /etc/apache2/httpd.conf

```
...
# Location of the Shibboleth's HS
<Location /shibboleth-idp/SSO>
     AuthType Fame
     AuthName "Fame Authentication Service"
     PerlAuthenHandler Apache2::Fame::sso_checker
     require valid-user
 </Location>
...
```

The second block specifies the location and settings for the F-LS (FAME Login Server). F-LS will direct a user's request to his/her chosen authentication server, and upon the successful authentication, generates an SSO cookie for the request session. The content of this block should look similar to the following:

File: /etc/apache2/httpd.conf

```
...
<Location /fame_login_server>
     SetHandler perl-script
     PerlResponseHandler Apache2::Fame::login_server
</Location>
...
```

The third block defines the (optional) FAME logout handler. If, for any reason, the user wishes to terminate his current SSO session without closing his Web browser, he may do so by clicking the "logout" button on the main FAME page. This will redirect the user to the FAME logout handler, configured below.

File: /etc/apache2/httpd.conf

```
...
<Location /fame_logout>
     SetHandler perl-script
     PerlResponseHandler Apache2::Fame::logout
</Location>
...
```

Fame.pm can be configured via a number of configuration parameters, as shown in the first two blocks above. These configurations can either be done in the Apache-Perl start-up script, *startup.pl* (as explained previously), using the module's *configure()* procedure, or in Apache's configuration file, *httpd.conf,* by defining each FAME parameter individually using the mod_perl's PerlSetVar directive. FAME's configuration parameters can be defined anywhere in the main body of *httpd.conf*, but outside the above three blocks. The following lists all FAME's configuration parameters.

File: /etc/apache2/httpd.conf

```
...
# FAME's configuration parameters
 PerlSetVar FameLoginServer /fame_login_server
 PerlSetVar FameLogoutHandler /fame_logout
 PerlSetVar FameDB dbi:mysql:database=fls;host=localhost;port=3306
 PerlSetVar FameAuthTimeout 1
 PerlSetVar FameSSOTimeout 480
 PerlSetVar FameDBuser flsuser
 PerlSetVar FameDBPassword flspassword
 PerlSetVar FameSecretsTable secrets:secret_key:date_time
 PerlSetVar FameASTable authservers:url:auth_type:loa:secret_key
 PerlSetVar FameUsersTable fameusers:ldap_id:ldap_attribute:alternativ_id
 PerlSetVar FameShibLDAPServer localhost
 PerlSetVar FameShibLDAPPort 389
 PerlSetVar FameShibLDAPDN cn=Manager,dc=example,dc=com
 PerlSetVar FameShibLDAPPassword your_secret
 PerlSetVar FameShibLDAPBaseDN ou=shib-users,dc=example,dc=com
...
```

Whichever of the two configuration methods you select to use (i.e. configuration via the *configure()* procedure in *startup.pl* or via the PerlSetVar directives in *httpd.conf*), note that the following precedence rules apply:

- If a directive is specified in both *httpd.conf* and the start-up script *startup.pl*, the value from the *httpd.conf* will be used and will override any other value.
- If a directive is neither specified in *httpd.conf* nor in *startup.pl*, the default value will be used, provided that the parameter has a default value. Otherwise, the module will report an error.

The following gives a detailed explanation of the FAME module's valid configuration parameters.

FameLoginServer

This parameter specifies the relative url of the Fame Login Server (F-LS), and corresponds to the second <Location> block specified in the *httpd.conf* above. This directive tells the F-SSO where to redirect the user if he/she has not been authenticated yet (detected by the absence of the **sso** cookie).

Default value: `/fame_login_server`.

FameLogoutHandler

This parameter specifies the relative url of the Fame Logout Handler. It is used to enable the users to logout and clear all FAME-set cookies, i.e. to reset the current SSO session.

Default value: `/fame_logout`.

FameAuthTimeout

This parameter specifies the duration (measured in minutes) during which a user is allowed to complete the authentication successfully. It is the time elapse from the moment when an auth-control cookie is created (by F-LS) to the moment when the auth-reply token is received (by the F-LS from the Authentication Server). If this timeout is expired before the user completes her/his authentication, the user will be forced to re-logon (i.e. re-authenticate).

Default value: 1 minute.

FameSSOTimeout

This parameter specifies the duration (measured in minutes) before the **sso** cookie is considered expired. In other words, it is the duration of a user's SSO session upon successful authentication. If this timeout is expired, the user will be forced to re-authenticate.

Default value: 480 minutes (8 hours).

FameDB

This parameter specifies the url string for the Perl DBI (DataBase Interface) to use when connecting to the FAME database. FAME has been developed using Mysql database, but a number of other database systems can be used with Perl DBI (see http://dbi.perl.org for details) and configured to work with FAME.

Format:
<*perl_database_inteface*>:<*database_type*>:database=<*database_name*>;host=<*host_name*>;port=<*port_number*>.

Default value: `dbi:mysql:database=fls;host=localhost;port=3306`.

If the host is 'localhost' and the port is '`3306`', they can be omitted from the FameDB string.

FameDBUser

This parameter specifies the *username* of the user who is allowed to connect the FAME to the FAME database above. The specified user must have read privileges for the FAME database.

Default value: `flsuser`.

FameDBPassword

This parameter specifies the password for the FameDBUser to use when making the connection to the FAME database.

Default value: `flspassword`.

FameSecretsTable

This parameter specifies the name of a table containing secret keys used by F-LS and F-SSO, as well as the names of the two columns of this table (one is used for the secret key itself and the other indicates the secret key version). Typically, a timestamp is used as a secret key version number when the key is inserted in the database.

Format:

<*table_name*>:<*secret_key_column_name*>:<*secret_key_version_column_name*>.

Default value: `secrets:secret_key:date_time`.

FameASTable

This parameter specifies the name for the table containing information about Authentication Servers, and the names of the five columns of this table. The five columns are: (1) the url of the Authentication Server, (2) the type of the Authentication Server (i.e. AuthType) that the Authentication Server provides, (3) the unique URN for the authentication method as defined by SAML1.1 (this is reserved for future extension; an SP may want to know the exact authentication method used, in addition to LoA), (4) the LoA of the Authentication Server, and (5) the secret key shared between the AS and F-LS (in Base64 format).

Format:
<*table_name*>:<*url_column_name*>:<*auth_type_column_name*>:<*saml_auth_id*>:<*loa_column_name*>:<*secret_key_column_name*>.

Default value:

`authservers:url:auth_type:saml_auth_id:loa:secret_key`.

FameUsersTable

This parameter specifies the name of the table that contains the mappings between the user identities used by FAME and their corresponding DNs stored in the Shibboleth LDAP directory. Each FAME user has a unique entry in the Shibboleth LDAP directory, identified by his LDAP DN, where the user's current LoA value is stored and picked up by Shibboleth. However, in addition, each FAME user may have several other types of identities as used by the Authentication Servers that the user has registered with, and the IdP may not use the LDAP directory to store the user's credentials. In this case, we need to map these different identities of the same user to the user's identity stored in the Shibboleth's LDAP directory. FameUsersTable contains these mappings. The table consists of three columns: the user's LDAP id (part of the user's LDAP DN), the name of the LDAP attribute for the user's id (e.g. *uid* or *cn*), and the user's alternative id (which can be the Kerberos principal name, *kerbuser@CS.MAN.AC.UK*, or the subject of the public key certificate, *C=GB/ST=Lancashire/L=Manchester/O=University of Manchester/OU=School of Computer Science/CN=Alex Nenadic/emailAddress=anenadic@cs.man.ac.uk*, or any other user's alternative username, depending on the Authentication Services used).

Format:
<*table_name*>:<*ldap_id_column_name*>:<*ldap_attribute_column_name*>:<*alternative_id*>.

Default value:

`fameusers:ldap_id:ldap_attribute:alternative_id`.

Note that, even if the IdP uses a LDAP for user authentication, all the LDAP users must be entered into the FameUsersTable. In this case, the <*ldap_id*> and <*alternative_id*> fields would contain the same values. As mentioned before, a single user may have multiple entries in this table, each containing the user's identity that the user has registered with a particular Authentication Server. However, all these entries must be linked to the user's identity used in the Shibboleth's LDAP directory.

FameShibLDAPServer

This parameter specifies the name (or IP address) of the LDAP Server Shibboleth uses for storing user attributes.

Default value: `localhost`.

FameShibLDAPPort

This parameter specifies the port number the Shibboleth LDAP Server is running on.

Default value: `389`.

FameShibLDAPDN

This mandatory parameter specifies the distinguished name of the user who manages the LoA attributes stored in the Shibboleth LDAP directory. The user must have the write privileges for the 'loa' attributes stored in the directory.

Example: `'cn=Manager,dc=example,dc=com'`.

Default value: `none`. The Fame module will attempt an anonymous binding if this item is not configured. It will almost certainly fail when an update to the 'loa' attribute is attempted later on, so it is strongly advised to set up this parameter.

FameShibLDAPPassword

This mandatory parameter specifies the password used by the FameShibLDAPDN.

Default value: `none`.

FameShibLDAPBaseDN

This mandatory parameter specifies the sub-tree of the Shibboleth LDAP directory where searches for the users should start from.

Example: `'ou=shib-users,dc=example,dc=com'`.

Default value: `none`.

2 FAME Database Configuration

The FAME database consists of three tables: the Secrets table, the Authentication Servers table, and the FAME Users table. A database user with read-only access to this database is required for the use by the Fame module. An exemplar database called *fls* and a database user *flsuser* (with password *flspassword*) can be created as follows.

```
# Create FAME database
CREATE DATABASE fls;
GRANT SELECT on fls.*
TO flsuser IDENTIFIED BY 'flspassword';
```

The names used for the database, for the Secrets, Authentication Servers and FAME Users tables, and for the database user and password can be, respectively, set up for the FAME module using configuration parameter *FameDB*, *FameSecretsTable*, *FameASTable*, *FameUsersTable, FameDBUser* and *FameDBPassword*.

Secrets table

The Secrets table stores the secret keys used by F-LS and F-SSO for creation of cookies (namely, the **sso** and the **request-url** cookies) passed between the two FAME components. Each key has a version number associated with it. The version number is simply the timestamp when the key was inserted into the database. We use this key version attribute to impose periodical updates of the secret key without invalidating current valid cookies generated with the key. This table is configured via parameter *FameSecretsTable* as follows.

```
# Create secrets table
CREATE TABLE secrets (
  secret_key text NOT NULL,
  date_time datetime NOT NULL default '0000-00-00 00:00:00'
);
```

Authentication Servers table

The Authentication Servers table stores information about configured/supported Authentication Server(s). Each row of the table has five columns: (1) the url at which the AS is running, (2) the authentication type (exemplar settings are Username/password, Kerberos, Browser certificate, Smart-card certificate) that are shown on the main FAME page as an option for the user to choose, (3) the authentication method identifier (reserved for use by future versions of the SAML protocol), (4) the LoA provided by the AS, (5) the secret key shared between the AS and the F-LS (which is used for encryption and decryption of the **auth-request** and **auth-reply** tokens), and (6) an optional timestamp. The last (i.e. timestamp) column in the table is optional, as it is not used by the FAME module. This table is configured via parameter *FameASTable* and can be created as follows.

```
# Create Authentication Servers table:

CREATE TABLE authservers (
  url varchar(100) NOT NULL default '',
  auth_type varchar(100) NOT NULL default '',
  saml_auth_id varchar(100) NOT NULL default NULL,
  loa smallint(6) NOT NULL default '1',
  secret_key text NOT NULL,
  date_time datetime NOT NULL default '0000-00-00 00:00:00',
  PRIMARY KEY  (url)
);
```

FAME Users table

The FAME Users table stores the mappings between the various identities that FAME users hold with different Authentication Servers and their identities used with the Shibboleth LDAP directory (in the Shibboleth LDAP directory, the users current LoA values are stored, which will be picked up later by Shibboleth IdP). Each row has three columns: (1) the user's LDAP id, (2) the name of the LDAP attribute that this id refers to (such as *uid*, or *cn*), and (3) the user's alternative id with an Authentication Server set up by the IdP. This table is configured via parameter *FameUsersTable* and can be created as follows.

```
# Example:

CREATE TABLE fameusers (
  ldap_id varchar(255) NOT NULL default '',
  ldap_attribute varchar(100) NOT NULL default '',
  alternative_id varchar(255) NOT NULL default ''
);
```

The *fame.sql* script found in the installation kit can be used to automatically create the FAME database called *fls* and the default database user *flsuser* (password *flspassword*). If you want to use a different username/password for the database user or different names for the database, table or column names (other than those used in

the script), modify the script to reflect this and make sure that you pass the correct values to the FAME module via the corresponding configuration parameters. To execute the *fame.sql* script, go to the directory where your *fame.sql* script is located and type in the following command.

```
$ mysql -u root -p < fame.sql
Enter password:
```

To verify that *fame* database has been created properly, connect to mysql database:

```
$ mysql -u root -p
Enter password:
```

Type in the administrator's password and you should receive the mysql prompt.

```
mysql>
```

Type in the following command to see the list of databases:

```
mysql> show databases;
```

You should be seeing something like the following:

```
mysql> show databases;
+------------+
| Database   |
+------------+
| fls        |
| mysql      |
| test       |
+------------+
5 rows in set (0.00 sec)

mysql> use fls;

Database changed
mysql> show tables;
+---------------+
| Tables_in_fls |
+---------------+
| authservers   |
| fameusers     |
| secrets       |
+---------------+
3 rows in set (0.00 sec)
```

3 Authentication Server(s) Configuration

FAME is designed to interoperate, and integrate, with existing authentication systems with minimum modifications. We here distinguish two cases of integration based on how an Authentication Server (AS) is implemented.

(1) If the AS is a custom-built system, then some modifications are required on the AS's side, so that the AS will receive requests sent by the F-LS component of the FAME system and return the requested information back to the F-LS upon user authentication. In this case, it is necessary to understand how information may be passed between the F-LS and the AS in order to make any necessary modifications to the AS. In other words, you need to work out how requests sent from the F-LS are received by your AS and how the responses sent by the AS are received by the F-LS upon a user's successful authentication. In cases where the authentication fails, the AS should return an error message to the user and ask for re-authentication.

An AS receives an authentication request from the F-LS in the form of:

https://<address_of_the_AS>?AuthRequestToken=<encrypted_auth_request_token>

This means that the encrypted **auth-request** token is passed to the AS as a url parameter. The **auth-request** token contains two parameters and is encrypted by the F-LS with a secret key, FLS_AS_KEY, shared between the F-LS and the AS, before it is passed to the AS. The AS needs to decrypt the token and extract the two parameters contained in it: a random challenge (*RC*) and the return address of the F-LS (i.e. the url address to which the user is re-directed to upon successful authentication). The two pareameters are separated by a comma (","). The format of the **auth-request** token is given below:

$$<auth\_request\_token> = <random\_challenge>,<address\_of\_the\_FLS>$$

$$<encrypted\_auth\_request\_token> = E_{FLS\_AS\_KEY}(< auth\_request\_token>)$$

Upon successful authentication, the AS is required to redirect the user back to the F-LS and pass the **auth-reply** token via URL, which looks like the following:

https://<address_of_the_FLS>/?AuthReplyToken=<encrypted_auth_reply_token>

The **Auth-reply** token is encrypted by the AS with the same shared symmetric key, FLS_AS_KEY. The **Auth-reply** token contains the AS's response to the random challenge received from the **auth-request** token (i.e. *RC*+1) and the username of the authenticated user, which will be later passed to Shibboleth (refer to Section 4.1 for information on what is considered as the username in this context), delimited by a colon (":"). The **auth-reply** token has the following format:

$$<auth\_reply\_token> = <random\_challenge\_response>:<user\_name>$$

$$<encrypted\_auth\_reply\_token> = E_{FLS\_AS\_KEY}(< auth\_reply\_token>)$$

(2) If an AS is a standard Apache module designed for protecting Web resources (e.g. the Kerberos system using Apache module *mod_auth_kerb* or an LDAP-based authentication system using *mod_auth_ldap*), then the integration between FAME F-LS and the AS is straight forward. You only need to install the script *AS.pm* provided in the FAME installation kit in order to make the FAME interoperate with the AS and no alterations to the AS are required.

In this case, the *AS.pm* script is used to get your AS to interoperate with the F-LS component of FAME. As *AS.pm* script is within the same namespace as *Fame.pm* module (i.e. *Apache2::* namespace), copy *AS.pm* to the same directory where *Fame.pm* is located (in this installation guide it is */etc/apache2/perl/MyApache2/*).

Let us suppose you are running Kerberos Authentication Server using Apache module *mod_auth_kerb*. You should create a <Location> in your *httpd.conf* that is tied to our *AS.pm* script and protected by Kerberos. Such a section in your *httpd.conf* may look something like the following:

File: /etc/apache2/httpd.conf

```
...
# Location protected by Kerberos
<Location /kerb-auth>
        SSLOptions +StrictRequire
        SSLRequireSSL
        PerlResponseHandler MyApache2::AS
        AuthType KerberosV5
        AuthName "Kerberos Login"
        KrbAuthRealms CS.MAN.AC.UK
        Krb5Keytab /etc/apache2/apache2_kerb.keytab
        KrbMethodK5Passwd on
        KrbServiceName HTTP
        KrbVerifyKDC on
        require valid-user
</Location>
...
```

The above code defines a <Location> within your Apache Web server served by the script *AS.pm.* As indicated by *https://<your_server>/kerb-auth,* the location is only accessible by a user using a Web browser upon a successfully authentication using the Kerberos AS over an SSL-protected connection. The <Location> */kerb-auth* plays the actual role of your AS. The *AS.pm* script serving this <Location> will receive requests from the F-LS (but only if the user has been authenticated by the defined AS), perform the necessary tasks, and redirect the user back to the F-LS.

The *AS.pm* script can be reused for any other authentication system (and not just Kerberos). The only thing that needs to be configured inside the script is the path to the file containing the Base64-encoded secret key shared between the F-LS and AS (look for variable $KEY_FH in the source code of *AS.pm* script).

4 FAME and Shibboleth Integration

After installing and configuring the FAME module by following the instructions given in Sections 1-3 above, the integration of the FAME module with the Shibboleth's IdP should proceed according to the following steps:

(1) Extend the Shibboleth LDAP directory's schema to include definitions of the two FAME-defined attributes: the 'loa' attribute and its expiration time;
(2) Insert the <SimpleAttributeDefinition> element in Shibboleth's *resolver.xml* to define the 'loa' attribute and the corresponding <JNDIDirectoryDataConnector> element to tell Shibboleth how and from which source to pull the loa attribute.
(3) Modify the Shibboleth IdP's ARP (Attribute Release Policy) located in file *site.arp.xml* to specify which requesting SPs should the *loa* attribute be released to.

Extend the Shibboleth's LDAP Directory Schema

The *inetOrgPerson* LDAP object class is widely used in LDAP directories to represents people within an organisation and has been endorsed by Shibboleth to store users' attributes in its LDAP store. The FAME system has extended this object class by defining a new LDAP object class called *famePerson* that inherits all attributes from the *inetOrgPerson* class and additionally defines two new attributes: *nist-loa* and *nist-loaExpires*. The first attribute is used to store the current 'loa' value for the user while the second stores the expiration time for the current 'loa' value (which is equal to the duration of an SSO session). The prefix '*nist*' in the attribute name is used to denote that the LoA definition used here is from the NIST E-Authentication Guideline[1]. The Shibboleth IdP is subsequently configured to pick up the *nist-loa* attribute from the LDAP store and pass it over to the requesting SP. The *nist-loaExpires* attribute is currently not passed to the SP, but is included in the LDAP store for potential future use.

The *famePerson* object class and the *nist-loa* and *nist-loaExpires* attributes are defined in the LDAP schema file called *fame.schema*, which can be found in the FAME installation kit.

File: fame.schema

```
# FAME LDAP schema.
# The attribute types and object class in this schema include the
# specifications of the 'loa' and 'loaExpires' attributes and the
# specification of the 'famePerson' object class. The 'famePerson'
# object class is an extension of the general purpose 'inetOrgPerson'
# object class, and additionally contains the two newly defined
# attributes 'loa' and 'loaExpires'.

# LoA attribute definition.
attributeType ( 1.2.826.0.1.3344810.1.1.104 NAME 'nist-loa'
        DESC 'The Level of Authentication Assurance conforming to the
NIST E-Authentication Guideline'
        EQUALITY integerMatch
        ORDERING integerOrderingMatch
```

[1] NIST Special Publication 800-63, E-Authentication Guideline, available at http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf

```
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        SINGLE-VALUE)

# LoA's expiration date attribute definition.
attributeType ( 1.2.826.0.1.3344810.1.1.106 NAME 'nist-loaExpires'
        DESC 'Expiration date for the current LoA attribute'
         EQUALITY caseExactMatch
         ORDERING caseExactOrderingMatch
         SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
         SINGLE-VALUE)

# famePerson object class definifion.
objectClass ( 1.2.826.0.1.3344810.1.0.24 NAME 'famePerson'
        DESC 'Person that uses FAME for authentication'
        SUP inetOrgPerson
        MAY ( nist-loa $ nist-loaExpires )
        )
```

To add this new attribute and object definitions to the IdP's LDAP directory, *fame.schema* has to be copied into the `schema` directory of the LDAP installation (on our system, it is `/etc/openldap/schema`), where other LDAP schemas are stored as well. Then the copied schema has to be included in the LDAP's configuration by inserting the line #4, as shown below, in the IdP LDAP Server's configuration file (typically `/etc/openldap/slapd.conf`):

File: `/etc/openldap/slapd.conf`

```
...
include         /etc/openldap/core.schema
include         /etc/openldap/cosine.schema
include         /etc/openldap/inetorgperson.schema
include         /etc/openldap/fame.schema
...
```

Configure Attribute Definitions and JNDI Data Connectors

The Shibboleth IdP acquires all the attributes it sends to an SP by using a specialised *attribute resolver* defined in the file *resolver.xml*. In order for an attribute to be sent to the SP, it has to be converted to a SAML-based XML format and included in the *resolver.xml* file in the form of the <SimpleAttributeDefinition> element. Then, a <JNDIDirectoryDataConnector> element has to be defined, which will be referred to by the <SimpleAttributeDefinition> element just created for holding the 'loa' attribute, in order to tell Shibboleth how to pull the 'loa' attribute from a data store (in our case, the Shibboleth LDAP directory).

To define the <SimpleAttributeDefinition> element for the 'loa' attribute, insert the following in *resolver.xml*:

File: `/usr/local/shibboleth-idp/etc/`resolver.xml

```
...
<SimpleAttributeDefinition id="urn:oid:1.2.826.0.1.3344810.1.1.104"
sourceName="nist-loa">
                <DataConnectorDependency requires="directory"/>
</SimpleAttributeDefinition>
...
```

The above code defines the attribute whose unique URN-like name is "urn:oid:1.2.826.0.1.3344810.1.1.104" (derived from the nist-loa attribute's unique oid). A DataConnector with an id `"directory"` is used to obtain the attribute value. To define such a DataConnector, insert the following in *resolver.xml*:

File: /usr/local/shibboleth-idp/etc/resolver.xml

```
...
<JNDIDirectoryDataConnector id="directory">
        <Search filter="uid=%PRINCIPAL%">
                <Controls searchScope="SUBTREE_SCOPE"
returningObjects="false" />
        </Search>
        <Property name="java.naming.factory.initial"
value="com.sun.jndi.ldap.LdapCtxFactory" />
        <Property name="java.naming.provider.url" value=
"ldap://rpc56.cs.man.ac.uk/dc=rpc56,dc=cs,dc=man,dc=ac,dc=uk" />
        </JNDIDirectoryDataConnector>
...
```

This DataConnector element has an id=`"directory"`, connects to an LDAP directory defined by the url `"://rpc56.cs.man.ac.uk/"` and the LDAP directory root `"dc=rpc56,dc=cs,dc=man,dc=ac,dc=uk"`, and uses a search filter `"uid=%PRINCIPAL%"` to search for the users when searching for the `%PRINCIPAL%`'s (i.e. user's) attributes. Modify the <Property> element to correspond to your LDAP server's settings.

Configure Shibboleth IdP's Attribute Release Policy

The Shibboleth's ARP (Attribute Release Policy) determines which of the defined attributes finally gets released to which requesting SPs. It acts as a filter for the attributes stored in the LDAP directory – ARPs can only be used to release the attributes that are already stored in the LDAP directory and defined in *resolver.xml*; it can be used to limit what information gets released to whom. On the other hand, the attribute must be both defined in *resolver.xml,* and specified in the site's ARP in order for it to be passed to a requesting SP via Shibboleth.

The simplest configuration for the *loa* attribute is to define a site policy in *arp.site.xml* file. Policies stored in this file apply for the whole IdP's site, i.e. for every user for whom this IdP retrieves/releases information. In order to configure a simple policy to release the *loa* attribute to every requesting SP, the *arp.site.xml* should look like the following:

File: /usr/local/shibboleth-idp/etc/arps/arp.site.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<AttributeReleasePolicy
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:mace:shibboleth:arp:1.0"
xsi:schemaLocation="urn:mace:shibboleth:arp:1.0 shibboleth-arp-
1.0.xsd">
        <Description>Simplest possible ARP.</Description>
        <Rule>
                <Target>
                        <AnyTarget/>
                </Target>
                <!-- Loa Attribute -->
```

```
                <Attribute name="urn:oid:1.2.826.0.1.3344810.1.1.104>
                        <AnyValue release="permit"/>
                </Attribute>
        </Rule>
</AttributeReleasePolicy>
```